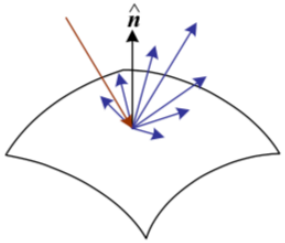
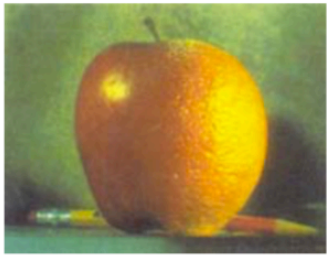
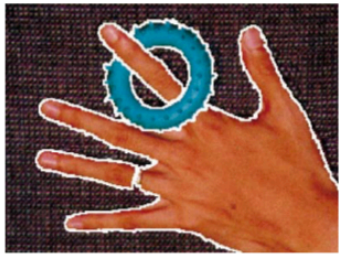


2. Image Formation



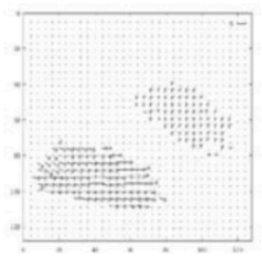
3. Image Processing

4. Features

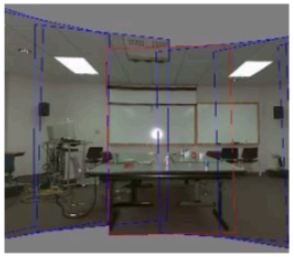


5. Segmentation

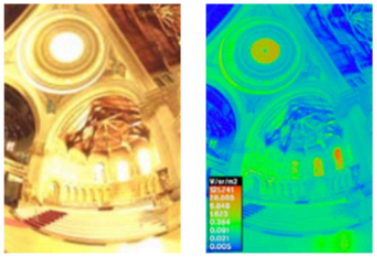
6-7. Structure from Motion



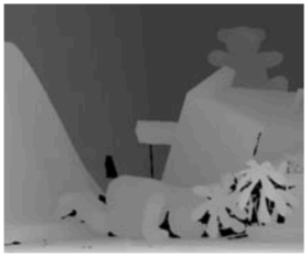
8. Motion



9. Stitching

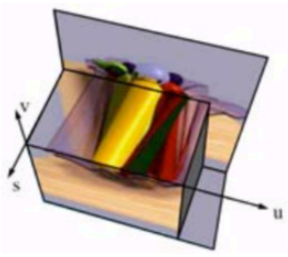


10. Computational Photography



11. Stereo

12. 3D Shape



13. Image-based Rendering

14. Recognition

Credits

- Images and formulas from Szeliski
- Second half from CVPR talk by Zhaoyang Lv

  Taking a Deeper Look at the Inverse Compositional Algorithm, Zhaoyang Lv , Frank Dellaert, James M. Rehg, Andreas Geiger, CVPR 2019

Translational Alignment

- Shift image I_1 with respect to template I_0
- Before, feature-based error:

$$E_{\mathrm{LS}} = \sum_i \|\boldsymbol{r}_i\|^2 = \sum_i \|\boldsymbol{f}(\boldsymbol{x}_i; \boldsymbol{p}) - \boldsymbol{x}'_i\|^2 .$$

Translational Alignment

- Shift image I_1 with respect to template I_0
- Before, feature-based error:

$$E_{\mathrm{LS}} = \sum_i \|\boldsymbol{r}_i\|^2 = \sum_i \|\boldsymbol{f}(\boldsymbol{x}_i; \boldsymbol{p}) - \boldsymbol{x}'_i\|^2.$$

- Now, image-based error:

$$E_{\mathrm{SSD}}(\boldsymbol{u}) = \sum_i [I_1(\boldsymbol{x}_i + \boldsymbol{u}) - I_0(\boldsymbol{x}_i)]^2 = \sum_i e_i^2,$$

SSD

$$E_{\mathrm{SSD}}(\boldsymbol{u}) = \sum_i [I_1(\boldsymbol{x}_i + \boldsymbol{u}) - I_0(\boldsymbol{x}_i)]^2 = \sum_i e_i^2,$$

- Sum of Squared Differences
- Assumes: brightness constancy
- If u fractional: interpolation needed
  - Bilinear (fast, good)
  - Bicubic (slower, slightly better)

Robust Error Metrics

$$E_{\mathrm{SAD}}(\boldsymbol{u}) = \sum_i |I_1(\boldsymbol{x}_i + \boldsymbol{u}) - I_0(\boldsymbol{x}_i)| = \sum_i |e_i|.$$

- Quadratic error is unforgiving!
- Absolute error (SAD): allows for outliers
- Differentiable robust error metrics exist

Dealing with Boundary Conditions

- Should not count pixels outside
- Add two "window" functions
- Windowed SSD metric:

$$E_{\mathrm{WSSD}}(\boldsymbol{u}) = \sum_i w_0(\boldsymbol{x}_i) w_1(\boldsymbol{x}_i + \boldsymbol{u}) [I_1(\boldsymbol{x}_i + \boldsymbol{u}) - I_0(\boldsymbol{x}_i)]^2,$$

- Invariant to overlap: Root mean square:

$$A = \sum_i w_0(\boldsymbol{x}_i) w_1(\boldsymbol{x}_i + \boldsymbol{u}) \qquad RMS = \sqrt{E_{\mathrm{WSSD}}/A}$$

Violations of Brightness Constancy

- Estimate Bias and Gain

$$I_1(\boldsymbol{x}+\boldsymbol{u}) = (1+\alpha)I_0(\boldsymbol{x}) + \beta,$$

$$E_{\mathrm{BG}}(\boldsymbol{u}) = \sum_i [I_1(\boldsymbol{x}_i+\boldsymbol{u}) - (1+\alpha)I_0(\boldsymbol{x}_i) - \beta]^2$$

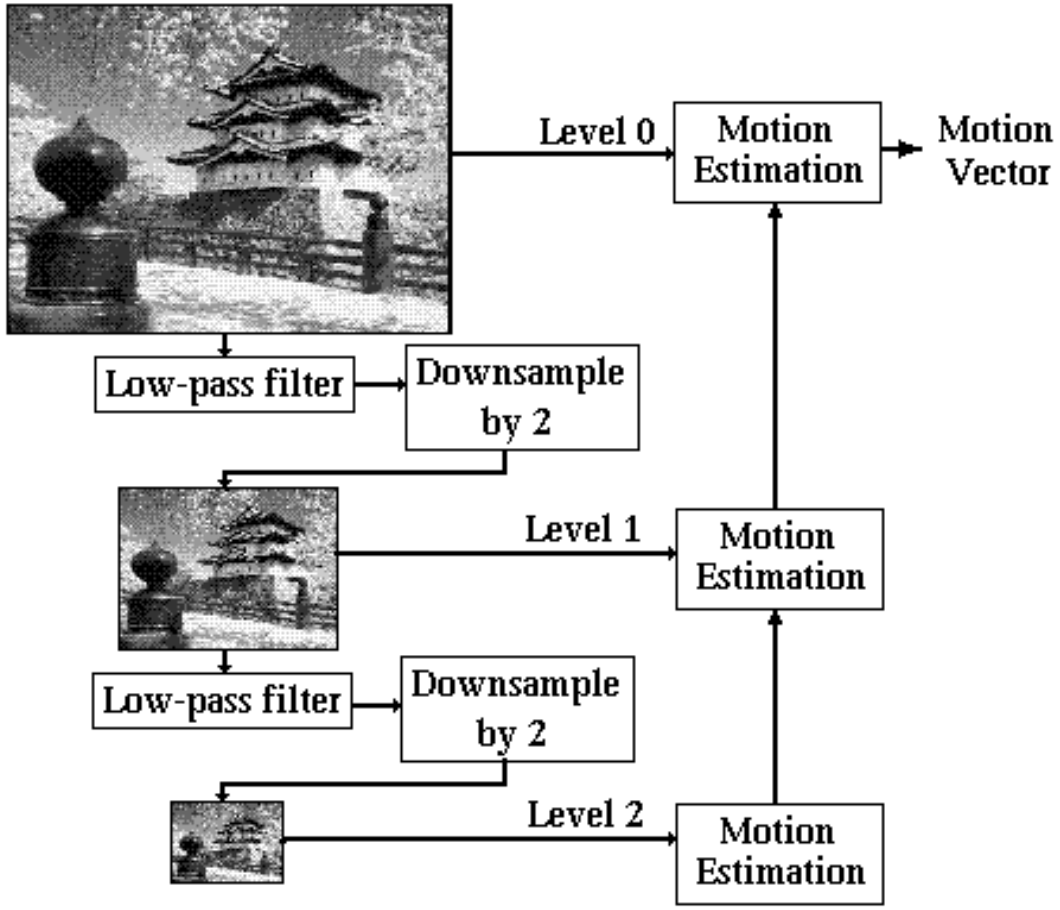
- Normalized Cross-Correlation

$$E_{\mathrm{CC}}(\boldsymbol{u}) = \sum_i I_0(\boldsymbol{x}_i) I_1(\boldsymbol{x}_i+\boldsymbol{u}).$$

$$E_{\mathrm{NCC}}(\boldsymbol{u}) = \frac{\sum_i [I_0(\boldsymbol{x}_i) - \overline{I_0}]\ [I_1(\boldsymbol{x}_i+\boldsymbol{u}) - \overline{I_1}]}{\sqrt{\sum_i [I_0(\boldsymbol{x}_i) - \overline{I_0}]^2}\sqrt{\sum_i [I_1(\boldsymbol{x}_i+\boldsymbol{u}) - \overline{I_1}]^2}}$$

Hierarchical Motion Estimation

- Build an image pyramid:
  - Low-pass
  - Decimate
- Recursively estimate motion:
  - Estimate motion at highest level
  - Use result as initial estimate at lower level



Sub-pixel Refinement

- Taylor expansion:

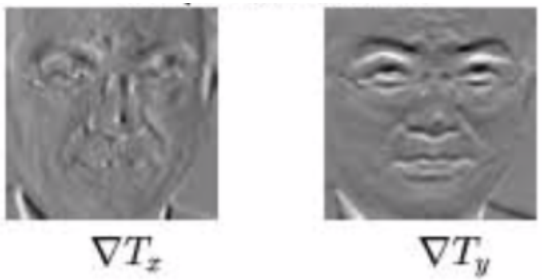
$$E_{\mathrm{LK-SSD}}(\boldsymbol{u}+\Delta\boldsymbol{u}) = \sum_i [I_1(\boldsymbol{x}_i+\boldsymbol{u}+\Delta\boldsymbol{u}) - I_0(\boldsymbol{x}_i)]^2 \quad (8.33)$$

$$\approx \sum_i [I_1(\boldsymbol{x}_i+\boldsymbol{u}) + \boldsymbol{J}_1(\boldsymbol{x}_i+\boldsymbol{u})\Delta\boldsymbol{u} - I_0(\boldsymbol{x}_i)]^2 \quad (8.34)$$

$$= \sum_i [\boldsymbol{J}_1(\boldsymbol{x}_i+\boldsymbol{u})\Delta\boldsymbol{u} + e_i]^2, \quad (8.35)$$

where

$$\boldsymbol{J}_1(\boldsymbol{x}_i+\boldsymbol{u}) = \nabla I_1(\boldsymbol{x}_i+\boldsymbol{u}) = (\frac{\partial I_1}{\partial x}, \frac{\partial I_1}{\partial y})(\boldsymbol{x}_i+\boldsymbol{u}) \quad (8.36)$$



Solve using Normal Equations

$$\boldsymbol{A}\Delta\boldsymbol{u} = \boldsymbol{b}$$

$$\boldsymbol{A} = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \quad \text{and} \quad \boldsymbol{b} = -\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

$$= \sum_i \boldsymbol{J}_1^T(\boldsymbol{x}_i + \boldsymbol{u})\boldsymbol{J}_1(\boldsymbol{x}_i + \boldsymbol{u})$$

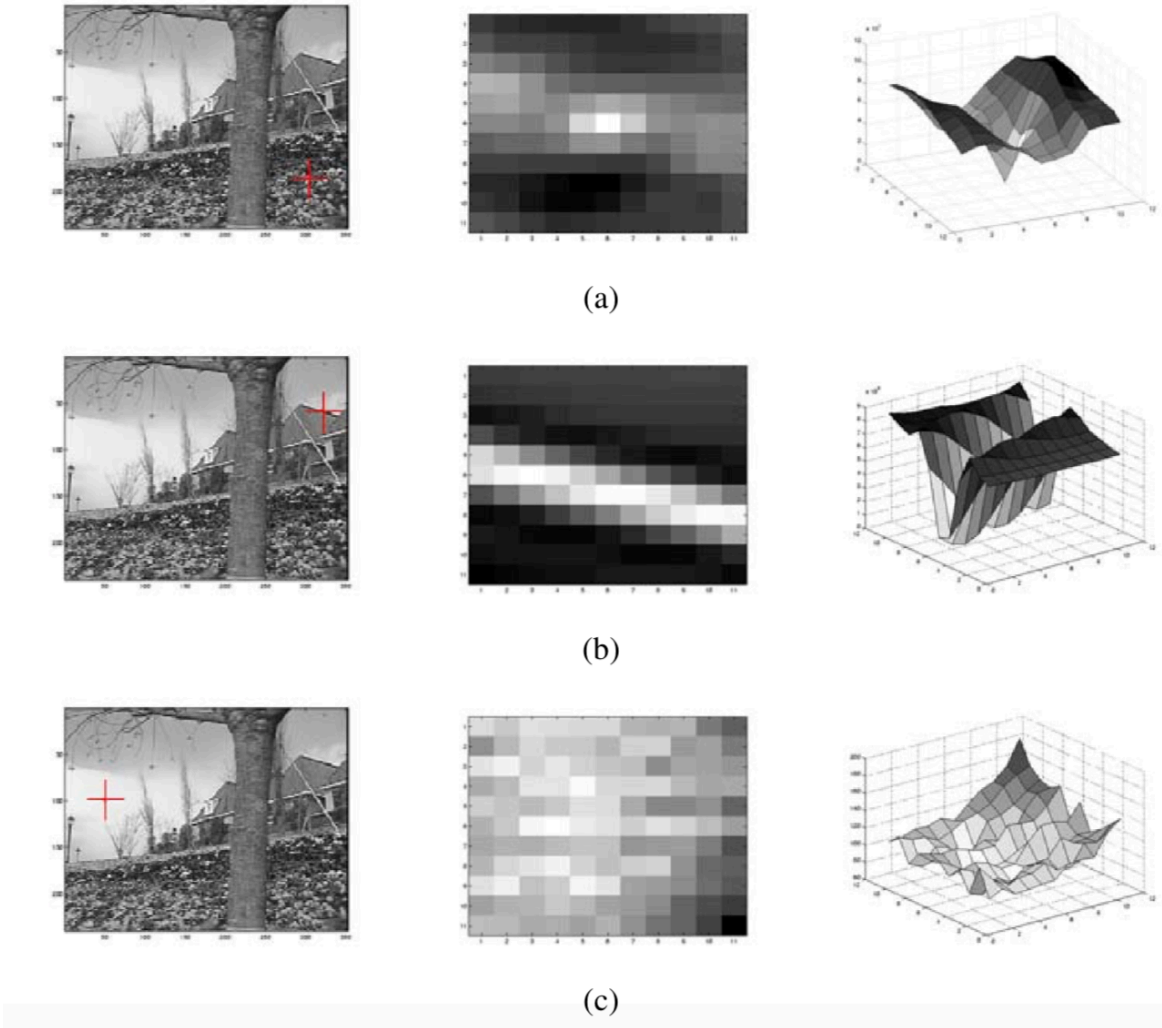
- RHS b is just dot product of gradient images with error ->
- A is Hessian or ”information matrix”, same as Harris uses!
- Remember: feature-based translation: just mean of flow vectors !

∇T_x

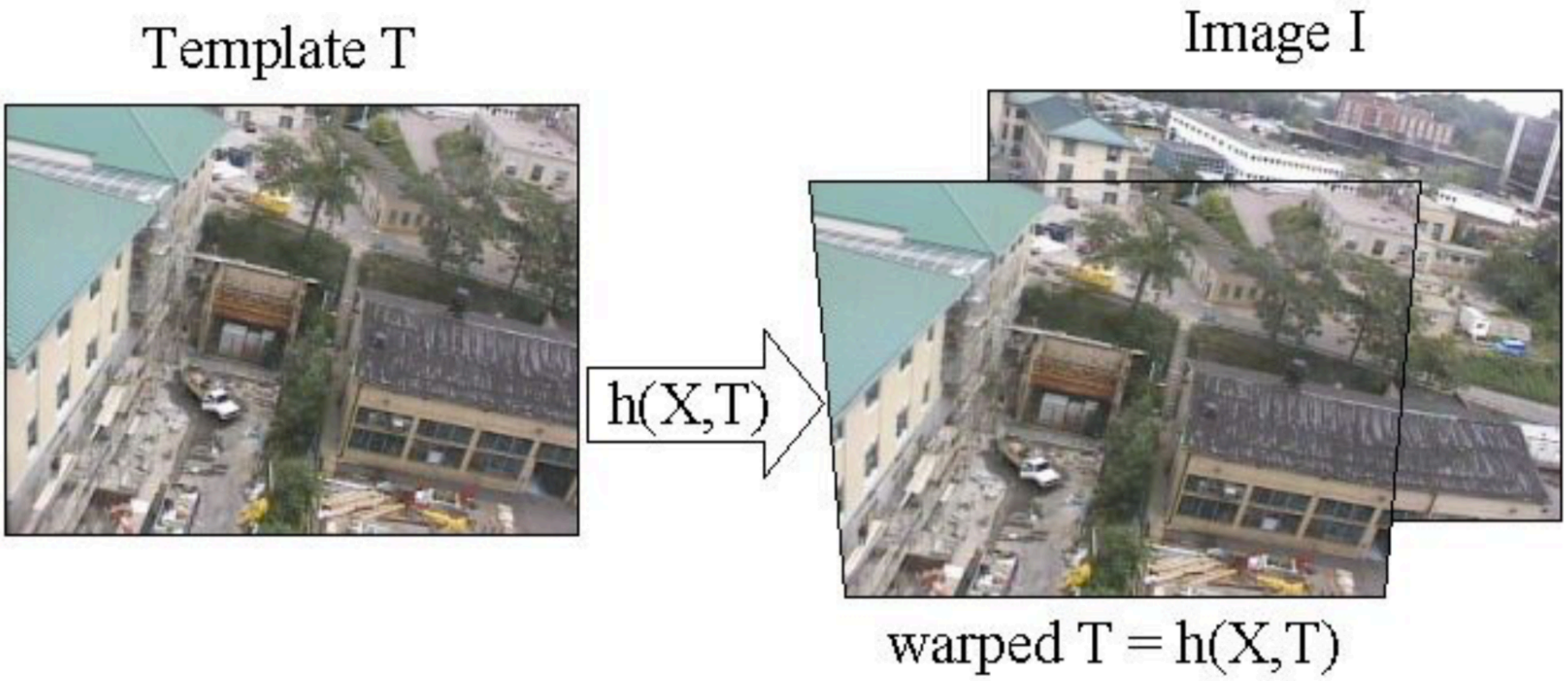
∇T_y

Error

Aperture Problems and Harris



Parametric Motion

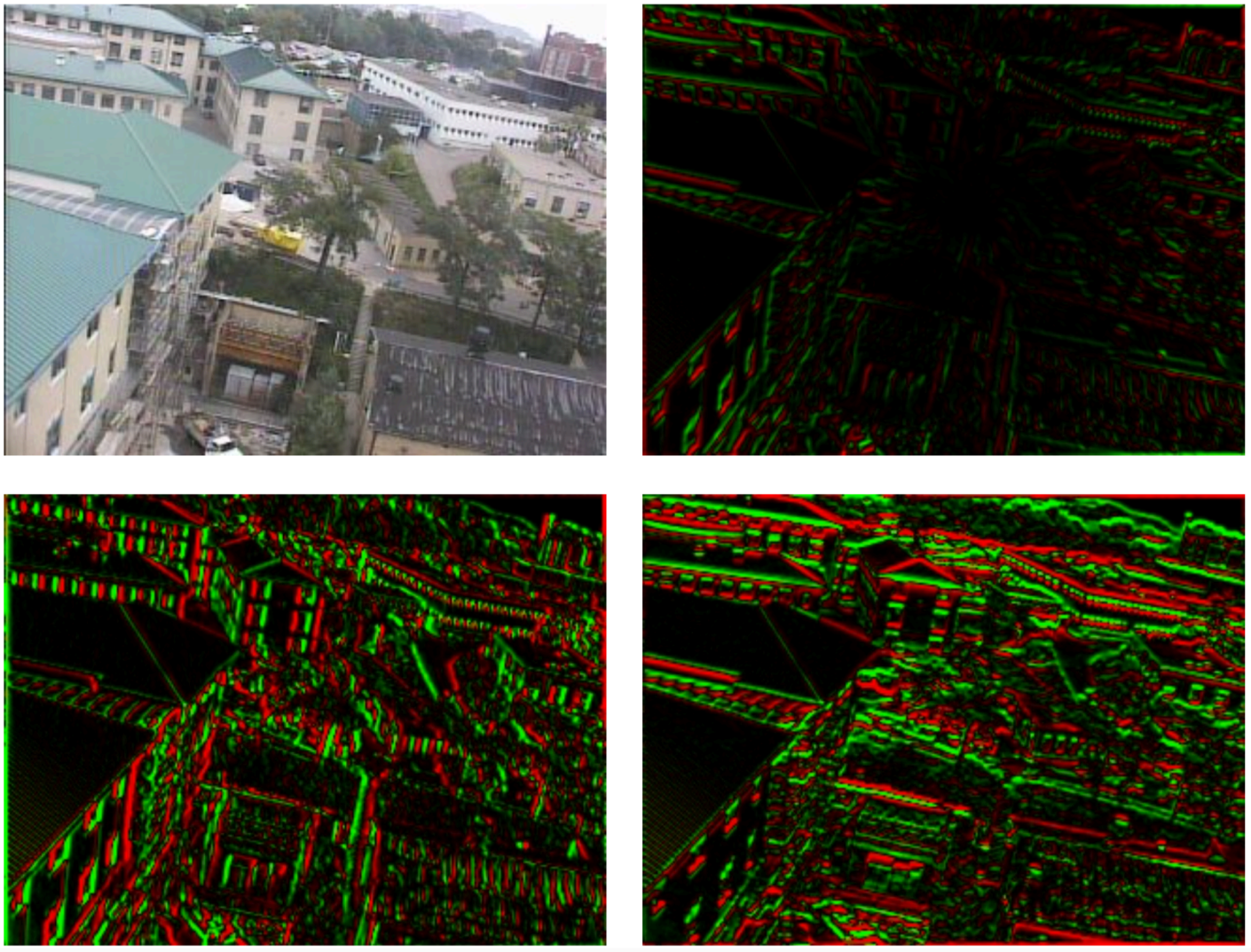


$$X = [a\ b\ c\ d\ e\ f\ g\ h]'$$

- E.g., image-based homography estimation

Dellaert & Collins, 1999, Fast Image-Based Tracking by Selective Pixel Integration

”Jacobian Images”

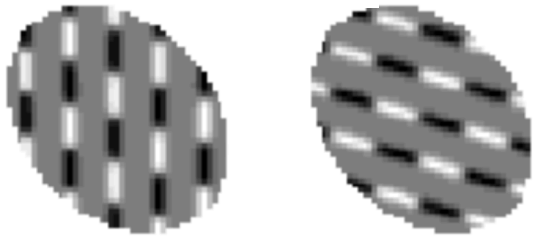


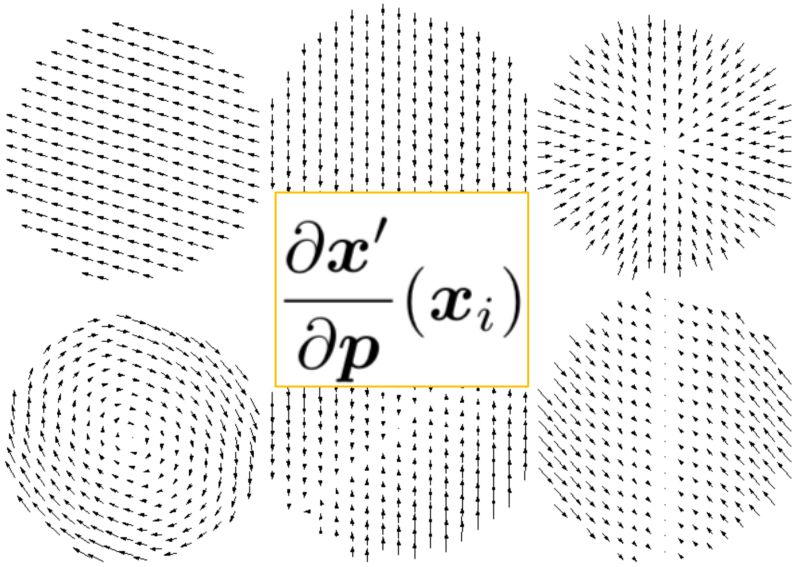
Dellaert & Collins, 1999, Fast Image-Based Tracking by Selective Pixel Integration

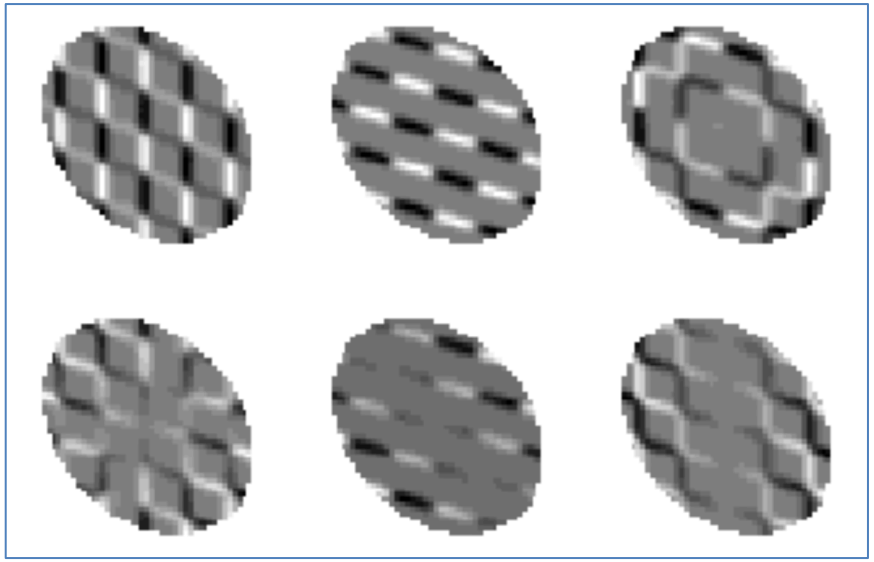
Computing Jacobian Images

template

$\nabla I_1(\boldsymbol{x}'_i)$



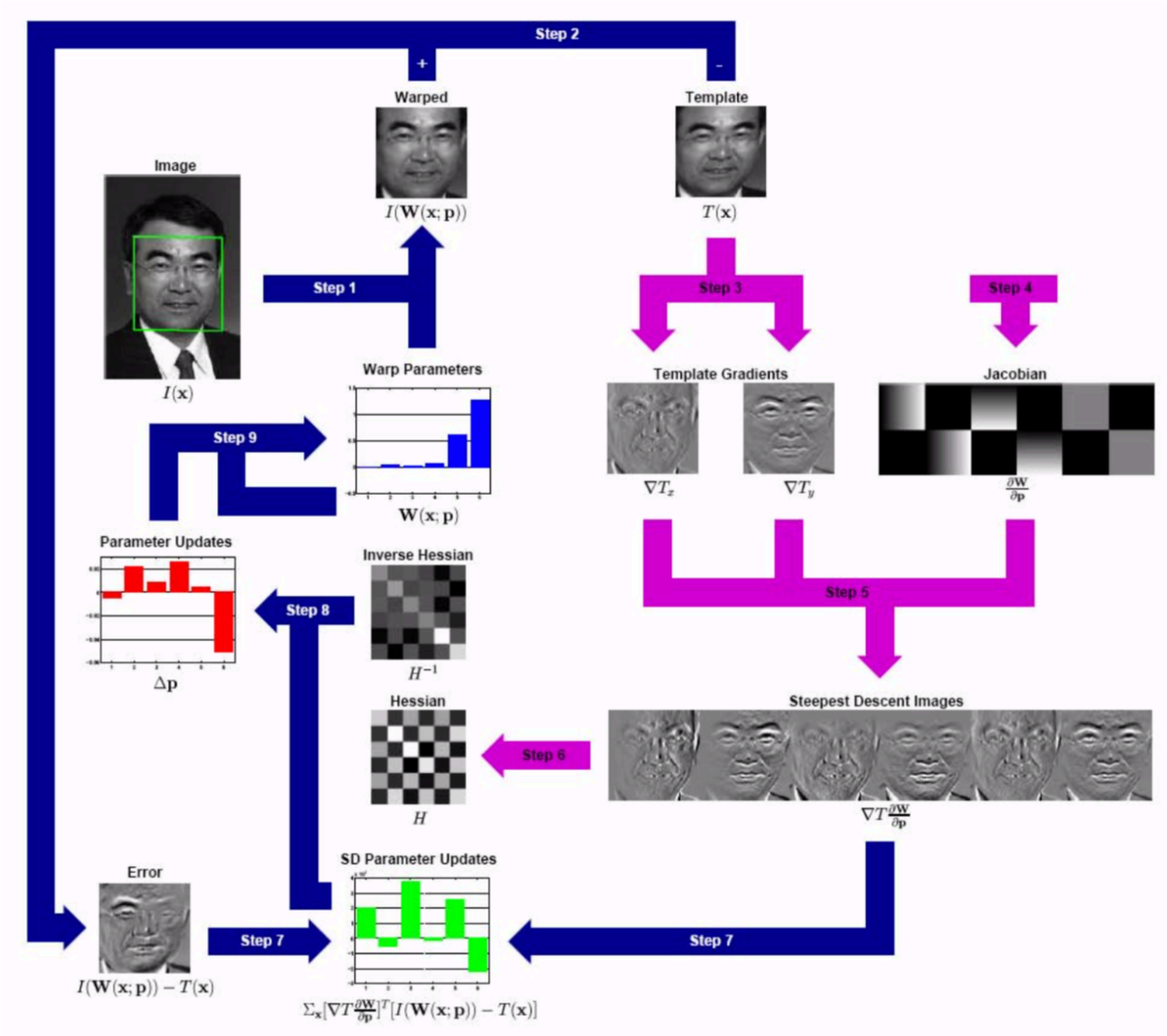




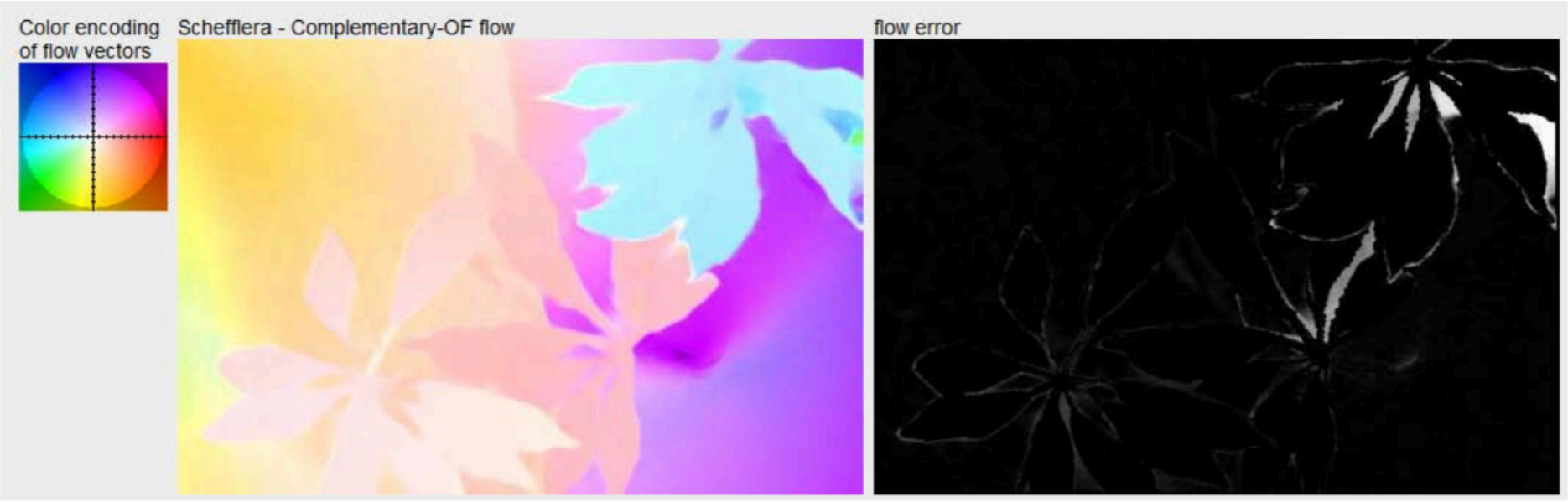
$$\boldsymbol{J}_1(\boldsymbol{x}'_i) = \frac{\partial I_1}{\partial \boldsymbol{p}} = \nabla I_1(\boldsymbol{x}'_i)\frac{\partial \boldsymbol{x}'}{\partial \boldsymbol{p}}(\boldsymbol{x}_i), \qquad (8.52)$$

Dellaert & Collins, 1999, Fast Image-Based Tracking by Selective Pixel Integration

Inverse Compositional Approach

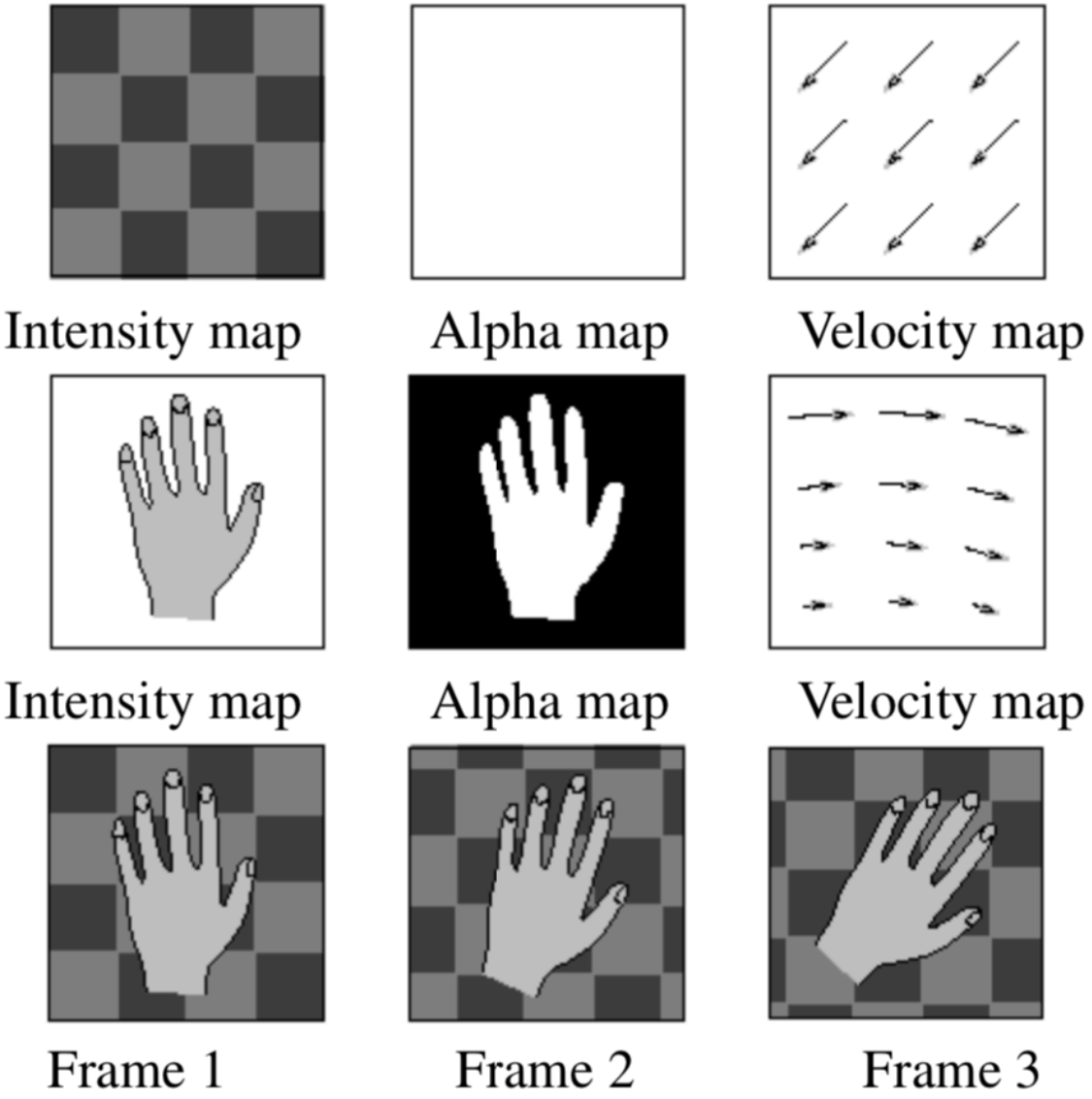


Optical Flow



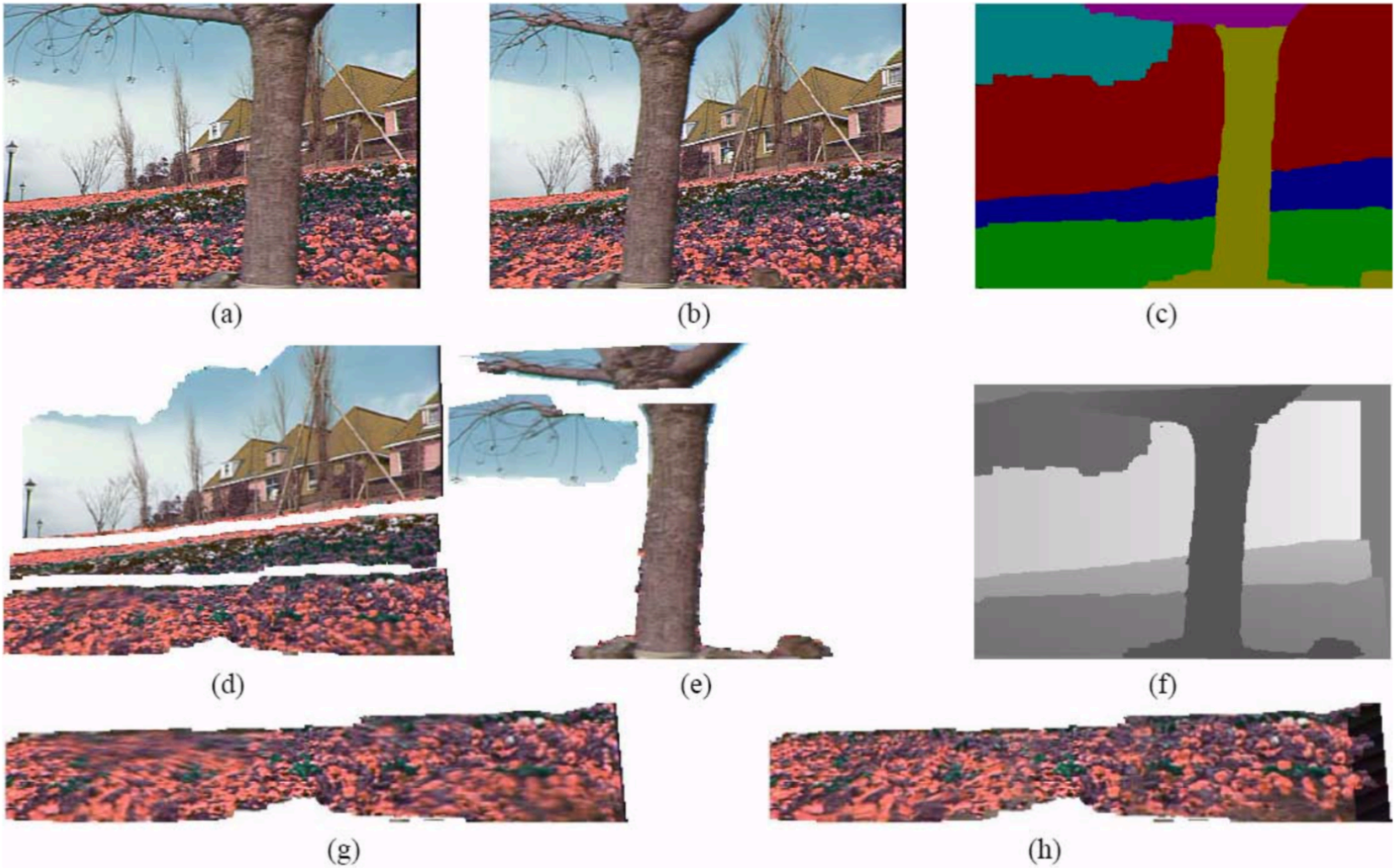
- Fully non-parametric model of motion
- N pixels -> N flow vectors -> 2N parameters
- Need some smoothness assumptions!
- Hard to deal with occlusion

Layered Motion



- One type of assumption to "regularize" optical flow
- Estimate FG and BG layers

Layered Motion Results



Baker, Szeliski, and Anandan 1998

The Inverse Compositional Algorithm [S. Baker and I. Matthews, 04]

We propose to take a **deeper** look at the Inverse Compositional algorithm **from a learning perspective.**

$$\mathbf{r}_k(\mathbf{0}) = \mathbf{I}(\boldsymbol{\xi}_k) - \mathbf{T}(\mathbf{0})$$

$$\Delta\boldsymbol{\xi} = (\mathbf{J}^T\mathbf{W}\mathbf{J} + \lambda\,\mathrm{diag}(\mathbf{J}^T\mathbf{W}\mathbf{J}))^{-1}\mathbf{J}^T\mathbf{W}\,\mathbf{r}_k(\mathbf{0})$$

$$\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k \circ (\Delta\boldsymbol{\xi})^{-1}$$

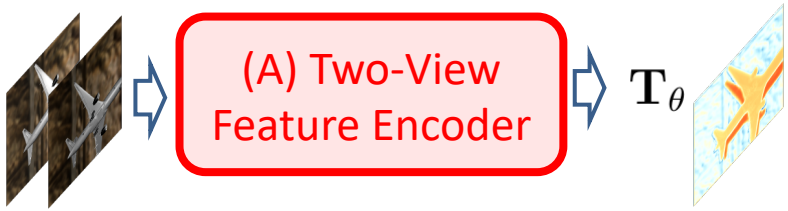
Take a Deeper Look at the Inverse Compositional algorithm

Contribution (A): Two-view Feature Encoder

$$\mathbf{r}_k = \mathbf{I}_\theta(\boldsymbol{\xi}_k) - \mathbf{T}_\theta(\mathbf{0})$$

$$\Delta\boldsymbol{\xi} = \left(\mathbf{J}^T\mathbf{W}\mathbf{J} + \lambda\,\mathrm{diag}(\mathbf{J}^T\mathbf{W}\mathbf{J})\right)^{-1}\mathbf{J}^T\mathbf{W}\,\mathbf{r}_k(\mathbf{0})\ ;$$

$$\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k \circ (\Delta\boldsymbol{\xi})^{-1}$$



Take a Deeper Look at the Inverse Compositional algorithm

Contribution (B): Convolutional M-estimator

$$\mathbf{r}_k = \mathbf{I}_\theta(\boldsymbol{\xi}_k) - \mathbf{T}_\theta(\mathbf{0})$$

$$\Delta\boldsymbol{\xi} = (\mathbf{J}^T \mathbf{W}_\theta \mathbf{J} + \mathrm{diag}(\mathbf{J}^T \mathbf{W}_\theta \mathbf{J})^{-1} \mathbf{J}^T \mathbf{W}_\theta \mathbf{r}_k(\mathbf{0})$$

$$\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k \circ (\Delta\boldsymbol{\xi})^{-1}$$

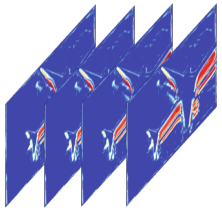
Take a Deeper Look at the Inverse Compositional algorithm

Contribution (C): Trust Region Network

$$\mathbf{r}_k = \mathbf{I}_\theta(\boldsymbol{\xi}_k) - \mathbf{T}_\theta(\mathbf{0})$$

$$\Delta\boldsymbol{\xi} = (\mathbf{J}^T\mathbf{W}_\theta\mathbf{J} + \mathrm{diag}(\boldsymbol{\lambda}_\theta))^{-1}\mathbf{J}^T\mathbf{W}_\theta\,\mathbf{r}_k(\mathbf{0})$$

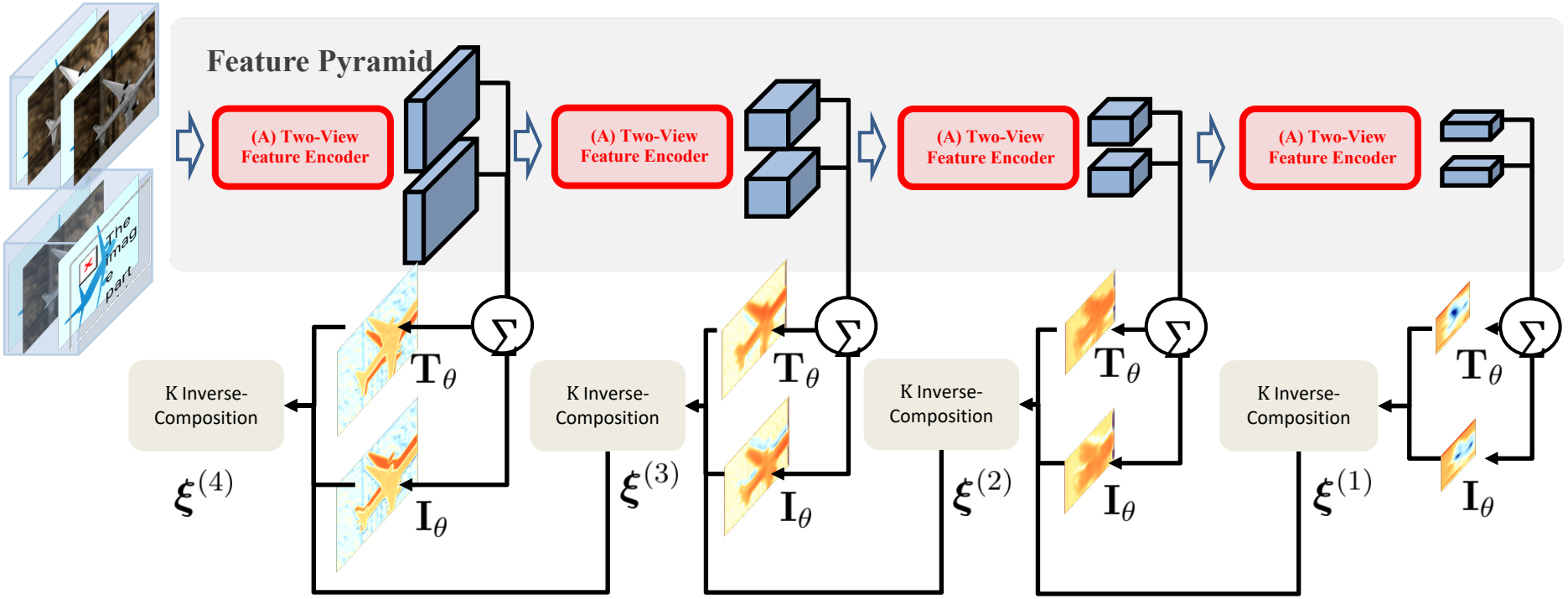
$$\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k \circ (\Delta\boldsymbol{\xi})^{-1}$$



(C) Trust Region Network

$\boldsymbol{\lambda}_\theta$

Coarse-to-Fine Inverse Compositional Algorithm



Conclusion

We have taken a deeper look at the inverse compositional algorithm by reformulating it with

(A) Two-view Feature Encoder

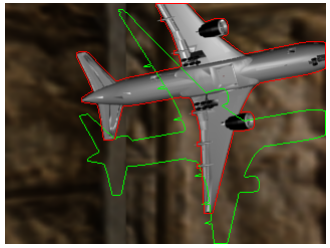
(B) Convolutional M-estimator

(C) Trust Region Network

The proposed solution is **learnable**, **accurate**, **small**, and **fast** in inference.

Visualization of Iterative 3D Rigid Motion Alignment

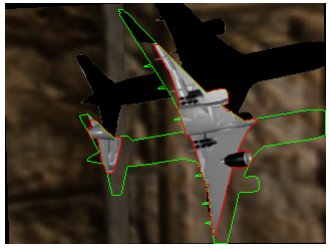
$\mathbf{T}$



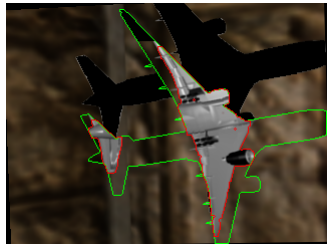
$\mathbf{I}$

$\mathbf{I}(\xi^{\mathrm{GT}})$

DeepLK
[Wang et al. ICRA, 2018]



Ours (A)



Ours (A)+(B)

Ours (A)+(B)+(C)